REMARKS

I. INTRODUCTION

In response to the Office Action dated August 12, 2003, claims 1-3, 5-7, and 9-11 have been amended. Claims 4, 8, and 12 have been canceled. Claims 1-3, 5-7, and 9-11 remain in the application. Entry of these amendments, and re-consideration of the application, as amended, is requested.

II. PRIOR ART REJECTIONS

In paragraphs (3)-(4) of the Office Action, claims 1, 2-5, 7-9, and 11-12 were rejected under 35 U.S.C. §103(a) as being unpatentable over Kish et al., U.S. Patent No. 5,890,176 (Kish) in view of Gerard et al., U.S. Patent No. 5,974,428 (Gerard). In paragraph (5) of the Office Action, claims 2, 6, and 10 were rejected under 35 U.S.C. §103(a) as being unpatentable over Kish in view of Gerard, as applied to claims indicated, and further in view of Cohen et al., U.S. Patent No. 6,324,543 B1 (Cohen).

Applicant respectfully traverses these rejections.

Specifically, claims 1, 5, and 9 were rejected as follows:

> As per claims 1, 5, 9, Kish teaches "obtaining a request to store an object" at col. 2, lines 20-34, col. 5, lines 4-21, col. 13, lines 46-54;
> "determining if a requested file version is lower than an object introduction version of the object" at col. 2, lines 20-44, col. 10, lines 42-59;
> "streaming out a class of the object in the requested file version if the requested file version is equal to or higher than the object introduction version; and streaming out the class of the object in the object introduction version if the requested file version is lower than the object introduction version" at col. 2, lines 26-29, col. 10, lines 14-16, col. 10, lines 45-51.

Applicant traverses the above rejections for one or more of the following reasons:

(1) Neither Kish, Gerard, nor Cohen teach, disclose or suggest streaming out a data representing an instance of an object;

(2) Neither Kish, Gerard, nor Cohen teach, disclose or suggest streaming out such data that includes/comprises actual methods and attributes of the object;

(3) Neither Kish, Gerard, nor Cohen teach, disclose or suggest saving a file by streaming out data representing the instance of the object in a requested file version if the requested file version is equal to or later than the object introduction version; and

(4) Neither Kish, Gerard, nor Cohen teach, disclose or suggest saving a file by streaming out data representing the instance of the object in the object introduction version if the requested file version is earlier than the object introduction version.

Independent claims 1, 5, and 9 are generally directed to storing object data in a particular version. Specifically, a file often contains objects. When the file is saved, the object (i.e., data representing an instance of the object) is streamed out/saved as part of the file. The amended claims also provide that the representative data comprises actual methods and attributes of the object. The claims further provide for storing a particular version of the object by streaming a data for a particular version of the object out to a file.

More specifically, the version of the file (that the object is being stored in) is compared to the version of the object when the object was introduced/originated. If the file version is the same or newer than when the object was first introduced, the file is saved such that (the data representing) the instance of the object is streamed out (e.g., stored) in the file version. However, if the file version is older than when the object was introduced, the file is saved such that the (data representing) the instance of the object is streamed out (e.g., stored) in the object introduction version. The cited references do not teach nor suggest these various elements of Applicant's independent claims.

The Office Action admits that Kish fails to teach the streaming steps. In this regard, Gerard is relied upon to teach these claimed elements. However, contrary to the assertion in the Office Action, Gerard fails to teach these claimed limitations.

In Gerard, a Java virtual machine uses a VersionLoader object which loads class files. The VersionLoader is passed a name of a desired class and returns a newly loaded class (see col. 5, lines 21-32). As illustrated in FIG. 4, when a request for a new object is received, a class must be loaded to support that request (see col. 8, lines 39-43). The request is sent to the VersionLoader object which utilizes a mapping mechanism (see col. 8, lines 43-47). The mapping mechanism supports a versioning policy that determines which class to return (see col. 8, lines 48-51).

The presently claimed invention provides for streaming out a particular version of an object. In this regard, the particular object that is streamed may be said to be based on a streaming policy (although Applicants disagree with referring to Applicant's invention as a "policy"). However, such an alleged particular policy in the present invention is clearly distinguishable and is not even remotely suggested by the policies described in Gerard.

None of Gerard's versioning policies teach, disclose, or suggest the presently claimed invention. In the present claims, an object introduction version is streamed out (to a file) if the file version is lower than the object introduction version. Further, an instance of the object is streamed out (to the file) in the file version if the file version is equal to or higher than the object introduction

version. Gerard fails to describe any such policy. In col. 7, lines 10-15, Gerard describes a policy where if a user requests a certain specific version of a class, that particular version is provided. Additionally, if a default version of the class is specified, the most current production version of the class is provided (see col. 7, lines 15-18). Accordingly, Gerard describes a policy that provides a particular version of a class if requested and otherwise provides the most recent version of the class. Gerard further describes the use of an abstract versioning policies in col. 7, lines 37-44. However, once again, Gerard fails to describe the particular streaming set forth in the present independent claims. Additionally, nowhere in Gerard is there any description of an object introduction version or a requested file version as claimed.

The Office Action relies on col. 8, lines 39-61 to teach the streaming out as claimed. However, col. 8, lines 39-61 merely describes the use of the VersionLoader object and a versioning policy that is supported by a version mapping mechanism used by the VersionLoader object. In this regard, the mere presence and use of an abstract versioning policy does not even remotely describe the use of a particular "policy". Nor does it describe saving a file by streaming out an instance of an object in a particular version as claimed.

Further, Gerard merely provides a class to a user for loading. In this regard, Gerard does not describe the storage of an file or object by streaming out a class or a request to store an object. Storing/streaming out a class of an object is distinguishable from providing and loading a new object for a user.

In addition to the above, Gerard teaches away from the present invention. Specifically, in Gerard's background, cols. 2-3, lines 66-6, Gerard describes the problems with using a class hierarchy as utilized in the present invention. Gerard provides that the use of a class hierarchy is extremely inefficient and can measurably reduce system performance. As a result, Gerard, teaches a methodology that does not use such a class hierarchy. In this regard, Gerard utilizes a class versioning and mapping mechanism to cross reference a requested class, select the most recent or best version of the requested class, and return an object to he user that belongs to the appropriate class (see col. 3, lines 32-39). More specifically, Gerard describes the use of a naming system as illustrated in FIGS. 2-4 and described throughout the specification (see for e.g. col. 7, lines 33-37). Such a teaching teaches away from using the hierarchical class system and using the comparisons and streaming methodology as claimed.

In addition to the above, as described above, Applicants have amended the independent claims to provide that the "streaming" is part of saving a file in a requested file version. Further, the

streaming is to a file that is being saved. Claims 4, 8, and 12 originally provided that the request to store the object was initialized by saving the file containing the object. The final Office Action rejects claims 4, 8, and 12 were rejected as follows:

> As per claims 4, 8, 12, Kish teaches "obtaining a request to store an object is initialized by saving a file containing the object" at col. 2, lines 20-34, col. 7, line 62 to col. 8, line 16, col. 13, lines 46-54.

Col. 2, lines 20-34 merely describe making and saving a copy of an object. Such a making and saving is further described in col. 10, lines 35-60. Applicants address the differences with respect to the col. 2 and 10 text in detail below.

With respect to col. 7, lines 62 to col. 8, line 16, Kish's text merely describes how the object pointers that are a vital part of Kish's invention are processed when document objects are stored in non-volatile storage. Namely, the pointers are converted to a form that is meaningful in the non-volatile storage format (see col. 7, lines 62-65). However, such a teaching fails to teach, saving a file by streaming out the data that represents an instance of an object (which includes actual methods and attributes of the object) to a file. In this regard, converting pointers to a form (i.e., an object ID) and storing such pointers is not even remotely equivalent to saving a file by streaming out actual methods and attributes as object data that represents an instance of an object. Accordingly, these portions of Kish fail to teach the invention as claimed.

Col. 13, lines 46-54 merely provides that a dialog box displays the various versions that exists in a particular file. However, such language does not refer to how the versions are stored in the particular file. Nor does such language indicate that the data in the file comprises actual methods and attributes of the object. Nor does such language illustrate that the object data is saved when a file is saved. Instead, the cited language merely provides that a single file contains various different object versions. Such a teaching is not equivalent, nor does it teach or suggest, the invention as claimed.

In response to some of the above arguments, the final Office Action provides:

> The Examiner respectfully disagrees for the following reasons:
>
> Per (1), as discussed in the above rejection, Gerard teaches when a request for a new object is received, a class must be loaded to support that request (col. 8, lines 39-43), the request is sent to the VersionLoader object which utilizes a mapping mechanism (col. 8, lines 43-47), and the mapping mechanism support a versioning policy that determines which class to return (col. 8, lines 48-51). It is thus clearly shown by Gerard the step of streaming out a class of an object.
>
> Per (2), the Applicant's invention discloses a method for storing object data of a requested file in an object-oriented computer system. Similarly, Kish reference teaches an object-oriented document version tracking method wherein a single file holds multiple versions of a document composed of an interconnection of objects which themselves have versions and are stored in the file (see Abstract). When the document is changed by changing any of the interconnected objects, a check is first made to determine whether the object version is the same as the document version currently being edited (i.e., determining if a requested file version is lower than an object introduction

version of the object). If not (i.e., whether it is equal or higher), a copy of the object is made and saved (i.e., the file version is older than when the object was introduced), the objects then must be structure so that any time one of these methods is called, the object checks to determine whether it is the current version. In a preferred embodiment, checking the existing version is performed by a common method in the base class CEngineObject called CheckVersion (col. 10, lines 45-51). Therefore, any version of the document can be reconstructed by interconnecting the object versions which have a highest level which is equal to or less than the desired document version, so only objects which are changed are duplicated and copies of objects are only made (i.e., storing object data in a particular version when an object changes.

Although Kish teaches streaming out an object (col. 2, lines 20-34), Kish does not explicitly teach the step of "streaming out a class of the object". Gerard, however, teaches this limitation as discussed in (1). More specifically, Gerard teaches selecting the most recent or best version of the requested class and return an object to the user or requesting object that belongs to the approved class (col. 3, lines 36-46). It would have been obvious to one ordinarily skilled in the art at the time of the invention to combine the teachings of Kish with the teaching of Gerard to include "streaming out a class of the object" in order to provide a system to have multiple versions of the same class on-line at the same time and to create and use objects from different version of the same class, as taught by Gerard at col. 3, lines 39-42.

Applicants traverse the above responses.

With respect to the final Office Action argument re: (1), Applicants note that the Office Action merely states that a class is returned by Gerard. However, the claims specifically provide that the streaming out is to a file and is part of saving the file. In this regard, returning a class is not equivalent to streaming out data representing an instance of an object to a file. In fact, Gerard specifically teaches that the classes are loaded for use in a Java virtual machine (see col. 8, lines 39-62). Accordingly, Gerard teaches away from saving a file by streaming object instance data to a file.

With respect to the final Office Action argument re: (2), Applicants note that the characterization of what Kish teaches is not supported by the language within Kish. The language cited in the Office Action is in col. 10, lines 34-59). Kish basically teaches that before an object or data of the object is changed, Kish determines if the existing version of the object is current. Such a determination compares the present version of the object to the current document version. If the versions are different (i.e., the object is older than the current document), the object creates a copy of itself by creating a new version that is current. If the versions are not different, processing merely continues. Of particular note is that the current version of the object is used in the comparison and not the object introduction version (as claimed). Further, the current document version is used in the comparison and not a requested document version (as claimed).

The office Action equates Kish's "checking whether the object version is the same as the document currently being edited" to the prior claim step of "determining if a requested file version is lower than an object introduction version of the object". However, as stated above, a current object version is not equivalent to an object introduction version as claimed. The object

introduction version as used throughout the claims (and specifically defined in the specification [see page 10, lines 15-20]) is the version in which the object was (first) introduced. Accordingly, there is a clear distinction between the Kish's determination and that made in the claims.

The Office Action then provides that Kish's making and saving a copy of the object is equivalent to streaming out the object in the file version as claimed. However, in Kish, the "copying" refers to creating an object in the new version (see col. 10, lines 51-54). The changes may then be made to the new object. However, the creation of a new object is not equivalent and does not suggest streaming out data that represents an object instance to a file in a file version. As stated above, Kish does not describe any such streaming out of data. Further, making a copy of itself and creating a new version is not equivalent to examining a particular requested file version (e.g., from a user) and based on that requested file version, conducting a comparison and streaming out data in the file version based on the comparison.

The Office Action then equates Kish's comparison of the active document version to the current object version to the claimed comparison between a requested file version and an object introduction version. Again, as stated above, the active document version is not equivalent to the requested file version. Kish's "current" or "active document version" is the most recent document version and is not a file version requested (e.g., by a user) that is not concerned with whether file version is "current" or "active". Further, the current object version is not equivalent to the object introduction version. As sated above, the object introduction version is the version when the object was first introduced and is not necessarily the current version of the object in the active document. Accordingly, the comparison conducted in Kish is not equivalent to (nor does it suggest or teach) the claimed comparison.

The Office Action notes that in Kish, copies of objects are only made when an object changes. However, the Office Action also equates such a teaching to the previously claimed storing object data in a particular version. However, such a conclusion is not rational. Teaching the timing of when to create a copy of an object (as in Kish) cannot possibly be related to or suggest storing object data in a particular version of the object. In this regard, the Office Action is equating Kish's timing aspect to an actual storing step.

In addition to the above, the references actually teach away from Applicant's invention. For example, the combined references would teach detecting if an object is going to be changed (Kish), copying itself to create a new version of the object if the object is old (Kish), later examining a single file that various versions of the object (Kish), and then loading a particular version of the object for

use in a Java virtual machine (Gerard). However, such a teaching does not refer to the following claimed elements: a requested file version, an object introduction version, saving a file by streaming out object instance data, object instance data that comprises actual methods and attributes, or the storage of particular object versions as claimed.

Further, the various elements of Applicant's claimed invention together provide operational advantages over the systems disclosed in Kish, Gerard, and Cohen. In addition, Applicant's invention solves problems not recognized by Kish, Gerard, and Cohen.

Thus, Applicant submits that independent claims 1, 5, and 9 are allowable over Kish, Gerard, and Cohen. Further, dependent claims 2-3, 6-7, and 10-11 are submitted to be allowable over Kish, Gerard, and Cohen in the same manner, because they are dependent on independent claims 1, 5, and 9, respectively, and because they contain all the limitations of the independent claims. In addition, dependent claims 2-3, 6-7, and 10-11 recite additional novel elements not shown by Kish, Gerard, and Cohen.

IV. CONCLUSION

In view of the above, it is submitted that this application is now in good order for allowance and such allowance is respectfully solicited. Should the Examiner believe minor matters still remain that can be resolved in a telephone interview, the Examiner is urged to call Applicant's undersigned attorney.

Respectfully submitted,

GATES & COOPER LLP

Attorneys for Applicant

Howard Hughes Center
6701 Center Drive West, Suite 1050
Los Angeles, California 90045
(310) 641-8797

Date: October 14, 2003

By: _____
Name: Jason S. Feldmar
Reg. No.: 39,187